|  |
| --- |
| Карагандинский технический университет имени Абылкаса Сагинова  Кафедра ИВС  **Лабораторная работа №3**  По дисциплине: "Системы управления базами данных"  *(наименование дисциплины)*  Тема: Создание отношений и диаграммы базы данных. Ввод данных.    **Преподаватель:**  Нуртай М.Д.  *(фамилия, инициалы)*  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *(подпись) (дата)*  **Выполнил:**  ст.гр. ITM-20-1 Кондратков К.А.  *(фамилия, инициалы)*  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *(подпись) (дата)*  Караганды 2022 |

**Тема:** " Создание отношений и диаграммы базы данных. Ввод данных."

**Цель работы:** Изучить способы создания отношений и диаграммы отношений базы данных средствами СУБД MS SQL Server 2012.

**Индивидуальное задание**

1. Построить отношения между таблицами вашей базы данных
2. Построить диаграмму вашей базы данных
3. Ввести в ваши таблицы тестовые наборы данных (в родительские таблицы – не менее 10 записей, в дочерние – не менее 30). При необходимости откорректируйте созданные в лабораторной работе № 2 ограничения на вводимые данные

**Результат выполнения индивидуального задания:**

Скрипт создания БД:

create database Lab\_rabota\_3

create table passenger

( surname nvarchar(30),

passenger\_name nvarchar(30),

IIN INT primary key,)

create table pilots

( id\_pilot INT identity (1,1) primary key,

surname nvarchar(30),

pilot\_name nvarchar(30),)

create table airplans

( number\_airplan INT primary key,

id\_pilot Int

FOREIGN KEY (id\_pilot) REFERENCES pilots(id\_pilot))

create table ticket

( number\_airplan Int,

place Int,

IIN Int,

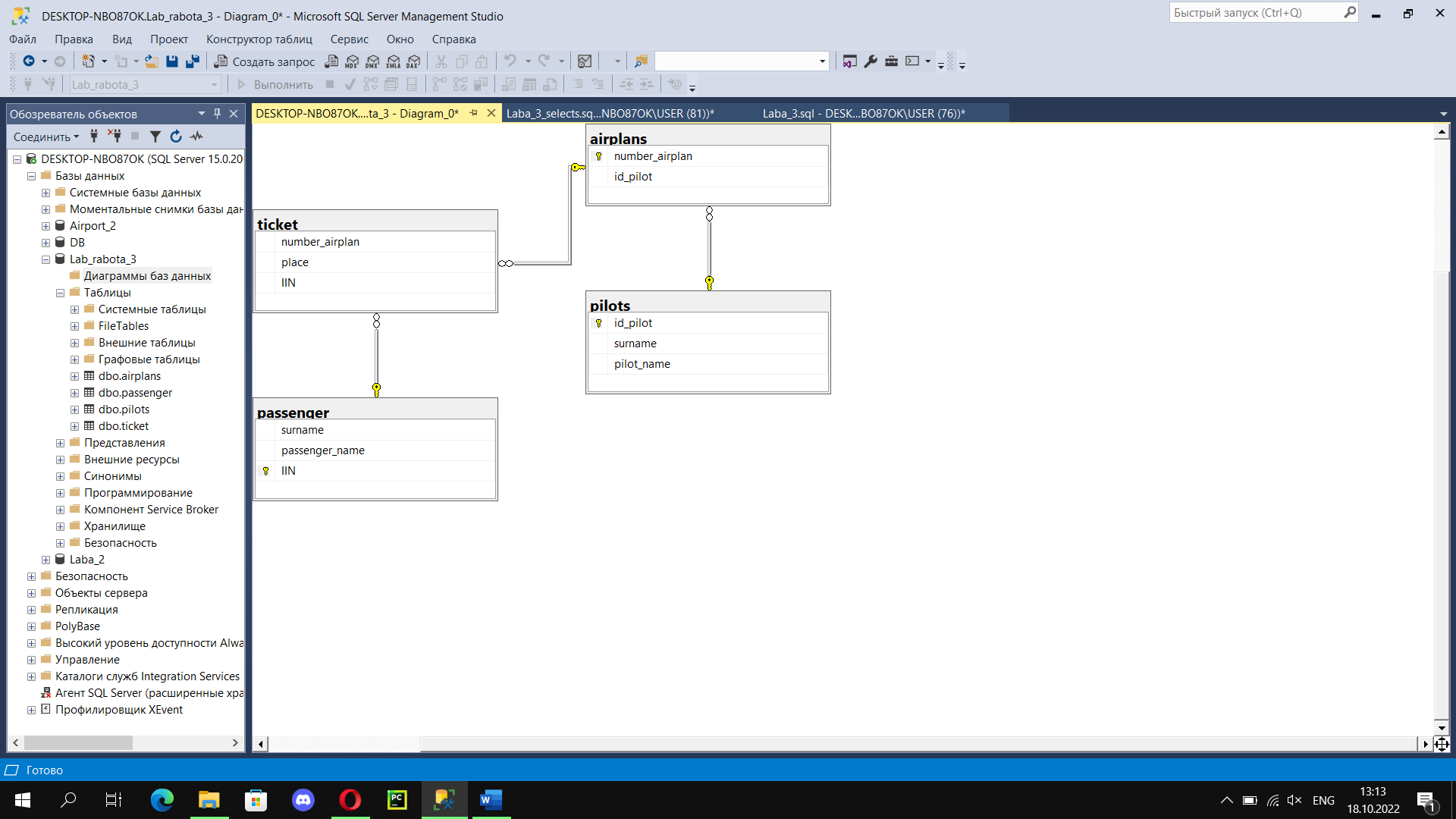
FOREIGN KEY (IIN) REFERENCES passenger(IIN),

FOREIGN KEY (number\_airplan) REFERENCES airplans(number\_airplan))

Скрипт вызова:

select passenger\_name,airplans.number\_airplan from passenger full join ticket on passenger.IIN = ticket.IIN full join airplans on airplans.number\_airplan = ticket.number\_airplan

Диаграмма базы данных:



**Ответы на контрольные вопросы:**

1. **Назначение механизма отношений. Виды отношений?**

Между двумя или более таблицами базы данных могут существовать отношения подчинённости. Отношения подчинённости определяют, что для каждой записи главной таблицы *{master,* называемой *еще родительской}* может существовать одна или несколько записей в подчинённой таблице *{detail,*называемой еще *дочерней}.*

Существует три разновидности связей между таблицами базы данных:

-        «один-ко-многим»,

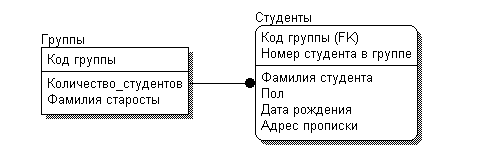
-        «один-к-одному»,

-        «многие-ко-многим».

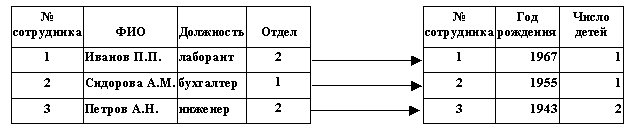
**Отношение «один-ко-многим»** имеет место, когда одной записи родительской таблицы может соответствовать несколько записей в дочерней таблице.

Связь "один-ко-многим" является самой распространенной для реляционных баз данных.

В широко распространенной нотации структуры баз данных IDEF1X отношение «один-ко-многим» изображается путем соединения таблиц линией, которая на стороне дочерней таблицы оканчивается кружком или иным символом. Поля, входящие в первичный ключ для данной ТБД, всегда расположены вверху и отчеркнуты от прочих полей линией.



**Отношение «один-к-одному»** имеет место, когда одной записи в родительской таблице соответствует одна запись в дочерней таблице.



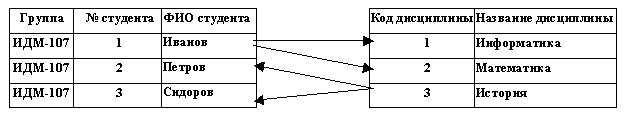
Данное отношение используют, если не хотят, чтобы таблица БД «не распухала» от второстепенной информации.

**Отношение «многие-ко-многим»** имеет место, когда:

а) записи в родительской таблице может соответствовать больше одной записи в дочерней таблице;

б) записи в дочерней таблице может соответствовать больше одной записи в родительской таблице.

Например, каждой студент изучает несколько дисциплин. Каждая дисциплина изучается несколькими студентами.



1. **Способы создания отношений**
2. **Ссылочная целостность данных. Способы ее поддержания?**

Ссылочная целостность относится к **взаимосвязи между значениями в логически связанных таблицах**. В реляционной модели это означает гарантирование логической непротиворечивости базы данных, за счет обеспечения постоянного соответствия значений первичного ключа и связанных с ним внешних ключей.

1. **Способы отображения зависимостей между таблицами БД**

Просмотр объектов, от которых зависит таблица

**Щелкните таблицу правой кнопкой мыши и выберите Просмотр зависимостей**. В диалоговом окне "Имя> объекта зависимостей<объектов" выберите объекты, зависящие от<имени> объекта, или объекты, от которых<зависит имя>объекта. Выберите объект в сетке Зависимости .

1. **Способы ввода данных в таблицы. Ограничения целостности при вводе данных**

Существуют два основных синтаксиса инструкции INSERT INTO:

INSERT INTO ИМЯ\_ТАБЛИЦЫ (столбец1, столбец2, столбец3,... столбец N)

VALUES (значение1, значение2, значение3,... значение N);

Здесь столбец1, столбец2, столбец3,… столбец N — это названия столбцов в таблице, в которые вы хотите вставить данные. Вы можете не указывать имя столбца в SQL-запросе, если добавляете значения для всех столбцов таблицы. Но убедитесь, что порядок значений соответствует порядку столбцов в таблице.

В таком случае синтаксис инструкции SQL INSERT INTO будет следующим:

INSERT INTO TABLE\_NAME VALUES (значение1, значение2, значение3,... значение N);

Ограничения UNIQUE

Ограничения — это правила, применяемые SQL Server ядром СУБД. Например, ограничения UNIQUE можно использовать для обеспечения того, чтобы в указанные столбцы, не входящие в состав первичного ключа, не вводились повторяющиеся значения. Хотя уникальность значений ограничения UNIQUE и PRIMARY KEY гарантируют в равной степени, в случае, когда необходимо обеспечить уникальность в столбце или комбинации столбцов, которые не являются первичными ключевыми, вместо ограничения PRIMARY KEY следует использовать ограничение UNIQUE.

В отличие от PRIMARY KEY, ограничения UNIQUE допускают значение NULL. Однако, как и всякое другое значение столбца с ограничением UNIQUE, NULL может встречаться только один раз. На ограничение UNIQUE могут ссылаться ограничения FOREIGN KEY.

При добавлении ограничения UNIQUE в существующий столбец или столбцы таблицы по умолчанию ядро СУБД проверяет существующие данные в столбцах, чтобы убедиться, что все значения уникальны. Если ограничение UNIQUE добавляется в столбец с повторяющимися значениями, ядро СУБД возвращает ошибку и не добавляет ограничение.

Ядро СУБД автоматически создает уникальный индекс для принудительного применения требования уникальности ограничения UNIQUE. Поэтому при попытке вставить повторяющуюся строку ядро СУБД возвращает сообщение об ошибке, которое указывает, что ограничение UNIQUE было нарушено и не добавляет строку в таблицу. Для обеспечения выполнения ограничения UNIQUE по умолчанию создается уникальный некластеризованный индекс, если явно не указано создание кластеризованного индекса.

Ограничения CHECK

Проверочные ограничения принудительно сохраняют целостность домена, ограничивая значения, которые может принимать один или более столбцов. Проверочное ограничение можно создать с любым логическим выражением, возвращающим значение TRUE или FALSE на основе логических операторов. Например, чтобы ограничить интервал значений столбца salary , можно создать ограничение CHECK, позволяющее столбцу принимать значения только в интервале от 15 000 до 100 000 долларов. Это ограничение исключает возможность устанавливать размер зарплаты, отличный от обычного. Логическое выражение будет иметь следующий вид: salary >= 15000 AND salary <= 100000.

К одному столбцу можно применять несколько проверочных ограничений. Кроме того, можно применять одно проверочное ограничение к нескольким столбцам. Для этого ограничение нужно создать на уровне таблицы. Например, с помощью проверочного ограничения, распространяющегося на несколько столбцов, можно подтвердить, что в любой строке со значением USA в столбце country\_region имеется также двухсимвольное значение в столбце state . Это позволяет выполнить проверку сразу нескольких условий из одного выражения.

Проверочные ограничения подобны ограничениям внешнего ключа, так как они управляют значениями, которые присваиваются столбцу. Однако они по-разному определяют допустимые значения: ограничения внешнего ключа получают список допустимых значений из другой таблицы, а проверочные ограничения определяют допустимые значения по логическому выражению.

1. **Удаление данных, удаление таблиц**

Удаление из базы данных происходит **с помощью команды "DELETE"** (переводится с английского как "УДАЛИТЬ"). Функция удаляет не одну строку, а несколько, при этом выбирает для удаления строки по логике функции "SELECT".

Для удаления таблицы из БД применяется команда **DROP TABLE**, после которой указывается название удаляемой таблицы. Например, удалим таблицу Clients:

DROP TABLE Clients;